THE CLAIMS

A detailed listing of all Claims 1-52 is provided below. A status identifier is provided for each claim in a parenthetical expression following each claim number.

1. (Previously Presented) A computer having a memory storing computer-executable instructions supporting plural objects and a mutation object, said mutation object comprising:

a method for mutating at least one object of said plural objects dynamically during run-time to provide a new implementation within the one object, wherein the one object includes a first method and a second method and an interface with a pointer, and

wherein the method of mutating includes changing the pointer from identification of the first method to identification of the second method.

2. (Previously Presented) The computer of Claim 1 wherein each one of said plural objects comprises:

a V-table;

a V-table pointer pointing to said interface.

3. (Canceled)

4. (Original) The computer of Claim 2 wherein said interface comprises a Mutate_Object method.

5. (Canceled)

6. (Previously Presented) The computer of Claim 2 wherein said mutation object mutates said V-table pointer so as to change the interface of the one object to a new interface corresponding to a new set of methods.

7. (Original) The computer of Claim 6 wherein said method of said mutation object is a Mutate_VTable method.

8. (Canceled)

9. (Previously Presented) The computer of Claim 2 wherein said method of said mutation object is a Mutate_Object method.

10. (Original) The computer of Claim 1 wherein each one of said plural objects comprises a state register storing a state of said one object, and wherein said method of said mutation object changes the contents of said state register so as to mutate the state of said one object.

11. (Original) The computer of Claim 10 wherein said state register stores the value of a pointer of said one object.

12. (Original) The computer of Claim 11 wherein said pointer of said one object comprises a VTable pointer.

13. (Canceled)

14. (Original) The computer of Claim 11 wherein said mutation object comprises a Mutate_Object method.

15. (Original) The computer of Claim 1 wherein said mutation object further comprises a synchronization of the mutation of one of said plural objects with threads running in said one object.

16. (Original) The computer of Claim 15 wherein said synchronization comprises mutual exclusion.

17. (Original) The computer of Claim 16 wherein said mutual exclusion prevents new threads from accessing said one object while other threads running in said object are permitted to finish.

18. (Original) The computer of Claim 15 wherein said synchronization comprises transactional synchronization.

19. (Original) The computer of Claim 18 wherein said transactional synchronization rolls back the threads currently running in the one object and then permits mutation of the object.

20. (Original) The computer of Claim 15 wherein said synchronization comprises swizzling.

21. (Currently Amended) The computer of Claim 20 wherein said swizzling comprises suspending threads running in said one object, mutating ~~said~~ the one object and modifying ~~the~~ states of the suspended threads in accordance with the mutation of the one object, and thereafter reactivating the suspended threads.

22. (Currently Amended) The computer of claim 21 wherein thread states are swizzled between clean points in the thread execution, whereby the ~~thread becomes~~ threads are suspended at a clean point.

23. (Original) The computer of Claim 1 wherein one of said plural objects comprises an interposition object formed by said mutation object mutating a particular one of said plural objects and a copied object at least nearly identical to said one particular object, said interposition object differing from said one particular object in that said one particular object has a pointer to said copied object and a method of interposition between threads seeking said one particular object and said copied object.

24. (Original) The computer of Claim 23 wherein said interposition method comprises a filter.

25. (Previously Presented) The computer of Claim 24 wherein said filter is a read-only filter.

26. (Original) The computer of Claim 24 wherein said filter provides access based upon the identity of the requesting thread.

27. (Original) The computer of Claim 23 wherein said copied object is a copy of the one particular object.

28. (Original) The computer of Claim 27 wherein said interposition object is formed by copying said one particular object and mutating the resulting copy while said copied object is said one particular object.

29. (Original) The computer of Claim 7 wherein said new implementation corresponds to a software upgrade.

30. (Original) The computer of Claim 7 wherein said new implementation is a higher speed I/O driver.

31. (Original) The computer of Claim 7 wherein said new implementation comprises recently loaded code.

32. (Previously Presented) The computer of Claim 1 wherein said new implementation comprises a different arithmetic algorithm.

33. (Previously Presented) The computer of claim 1 where said new implementation is a version of an algorithm where specific conditions are assumed to be true, where the version is mutated back to a version when the conditions are no longer true.

34. (Original) The computer of claim 33 wherein some of the parameters of the method are assumed to be constant.

35. (Original) The computer of claim 34 where the version is generated by a compiler through constant folding.

36. (Original) The computer of claim 33 where specific assumptions are made of the objects the method accesses.

37. (Original) The computer of claim 36 where the assumption is the location of an object.

38. (Original) The computer of claim 36 where the assumption is the value of a field of the state of the object.

39. (Original) The computer of claim 36 where the said version is generated through constant folding.

40. (Original) The computer of claim 36 where the said version is generated through inlining.

Claims 41-52: (Canceled)